REMARKS

Status of the Claims

- Claims 1-24 are pending in the Application after entry of this amendment.
- Claims 1-24 are rejected by Examiner.
- No claims are amended to overcome the cited art.

Claim Rejections Pursuant to 35 U.S.C. §103 (a)

Claims 1-3, 5-6, 8-18, and 21-23 stand rejected pursuant to 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,243,856 to Meyer et al. in (Meyer) view of U.S. Patent No. 6,738,975 to Yee et al. (Yee). Applicant respectfully traverses the rejection.

Meyers teaches a system that accommodates the transfer and translation of animated scenes at a user's computer. As taught by Meyer:

> "The present invention is directed toward a system and method for the efficient handling of animated scenes in an extensible environment. Preferably, in accordance with one embodiment of the invention, the invention is directed toward the efficient handling of animated scenes for the downloading and execution of the animated scenes in an interactive manner at a user's computer at a remote location." (col. 2 lines 24-31).
>
> "In accordance with one embodiment of the invention, the animated sequence is provided in a highly extensible animation language such that the sequence can easily be run on any of a variety of computers. Preferably, in this embodiment, the animated sequence is re-written utilizing one or more custom opcodes. These opcodes provide for efficient techniques for rendering the animated scenes on the user's computer." (col. 2 lines 38-45).
>
> "In accordance with another aspect of the invention, the animated sequence is divided into a plurality of pieces which are successively loaded onto the user's machine. These pieces are referred to as loads. The loads can include functionality as well as any data necessary to render a portion of the animated scene on the user's computer. Proper apportionment of functionality and data to the various loads allows for an efficient process for downloading the data to the user's computer." (col. 2 lines 50-55).

Thus, Meyer teaches a method to download an animated scene onto a user's computer by re-writing or converting the animation into opcodes. The animated sequence is divided up into multiple loads which are downloaded to the user computer. Each load can render a portion of the animation.

Meyer further teaches:

> "According to one embodiment for implementing this example, the sample animation may be provided in four loads. The first load includes only the functions and data necessary to render the initial animated scene on the user's machine."(col. 3 lines 10-13).
>
> "The scene definition language is compiled into opcodes for interpretation by the nanokernel. The opcodes and the nanokernel files are converted into a highly extensible language suitable for operation on a variety of end-user machines." (col. 3 lines 47-51).

Thus, Meyer teaches that the first load contains only the functions (opcodes) and data necessary to render the initial animated scene on the user's machine assuming the users machine has a nanokernal to interpret specific animation language opcodes that represent the animation.

Applicant notes that cache is used in Meyer for the purpose of executing the opcodes in an efficient manner. Meyer teaches:

> "In one embodiment, memory or a data cache is associated with at least some of the opcodes. In this embodiment, data relating to the execution of the opcodes is cached so that it can be re-used in subsequent performance of the same operation." (col. 15 line 66- col. 16 line 3).

Applicant notes that Claim 1 recites in relevant part:

A method of locating classes in a class path, the method comprising:
generating a cache of information relating to the classes in the class path;...
requesting a search of the class path via the wrapper; and

searching the cache to satisfy the requested search.

In the present Office Action on page 3, the Examiner states that a Claim 1 element is disclosed in Mayer. Page 3 of the present Office Action states the element and the Meyer citation as:

> "requesting a search of the class path and searching the cache to satisfy the requested search (searching the class path: col. 23, lines 15-30, col. 27, lines 8-38 and lines 55-67 and col. 28, lines 1-10)." (Office Action, page 3)

Yet, Applicant notes that this is not the entire Claim 1 element. The full element of Claim 1 is "requesting a search of the class path *via the wrapper*; and searching the cache to satisfy the requested search". Applicant disagrees that Meyer teaches the Claim 1 element because the context of Meyer is different and the aspect of requesting a search using the wrapper is not found in Meyer.

Addressing the citations found on page 3 of the present Office Action, Meyer teaches at col. 23 lines 15-30:

> "FIG. 16 is an operational flow diagram illustrating an example process by which unmarked nodes can be designated with a load number in accordance with one embodiment of the invention. In a step, 1604, the developer examines the tree and designates what the developer believes to be a minimum set of nodes which need to be defined to allow the remainder of the nodes to be determined unambiguously. In a step 1608, the compiler traverses the tree looking for undesignated nodes. When an unmarked node is located, its parent node is examined to determine the load number for that parent. This is illustrated by steps 1610 and 1612. If there is only one parent to the node in question, that node is assigned the load number of the parent and the operation continues to search for additional unassigned nodes. This is illustrated at steps 1614 and 1616 and by load line 1662." (col. 23 lines 15-30)

Applicant notes that nodes are the opcodes with attendant data such as color, shape, surface material, texture, lighting, reflectivity, transformation and so on. (see col 13 lines 50-

53 and col. 16 lines 63-65). So, Applicant concludes that col 23, lines 15-30 teaches a method where the nodes (opcode and the respective data for the opcode) are designated with a load number so that the opcodes can be sent to a user's computer to render an image using the nanokernal. Applicant submits that this does not teach the Claim 1 element of "requesting a search of the class path via the wrapper".

Addressing the next citation of page 3 of the present Office Action, Meyer at col. col. 27, lines 8-38 teaches:

> "As described above, one technique that is utilized to avoid having to recalculate the values each time a function is executed is to cache values and to reuse those values from the cache. One technique for implementing a cache is now described according to one embodiment of the invention. Generally speaking, as a scene graph is traversed and functions are executed, parameters for results of those functions are saved in a cache. In simple terms, because different executions of the function can yield different results, a different or unique cache is preferably created each time a function is called in a different way (i.e., from a different set of predecessors in the graph). Thus, if a previously obtained result is subsequently required, that result is available in one of potentially a plurality of caches. To illustrate, consider a simple example in which a function to used to color an object. Going one execution, the function colors the object red; the result which actually colors the object red is stored in a first cache. During subsequent execution of the function, if the function is to again color the object red, the function does not need to be executed again, instead the result can simply be retrieved from the cache and the processing can continue. If, on the other hand, the function this time colors the object yellow, a new cache is defined and the results of this execution of the function is stored in this new cache. As such, there are now two instances of the function results which can be utilized during subsequent operations. Simply writing the results of the operation coloring the object yellow over those stored in the first cache (the results of coloring the object red) would obviously render those results unavailable for subsequent operations in which the operation was again colored red. As such, such an approach is less desirable." (col. 27 lines 8-38).

Applicant submits that the above is a teaching that caching is used in Meyer to store attendant data values, such as those in the nodes for color, shape, surface material, texture, lighting, reflectivity, transformation and so on, such that the execution of the opcodes can progress more quickly if a data cache for the opcode color, shape, surface material, etc. is available. Applicant submits that this does not teach the Claim 1 element of "requesting a search of the class path via the wrapper".

Addressing the next citation from page 3 of the present Office Action, Meyer at col. col. 27 lines 55-67 and col. 28, lines 1-10 teaches:

> "In a step 1912, the scene graph traversed to reach the first USEFUNC node in the scene graph. In a step 1916, it is determined whether this USEFUNC node has been traversed before. Specifically, a cache for that USEFUNC is examined to determine whether this USEFUNC node has been traversed from the first context. If it is the first time that this USEFUNC node has been traversed from the present context, a new context for that USEFUNC is defined in a step 1920. More specifically, a new context is associated with the cache for that USEFUNC. In a step 1924, the function referred to by that USEFUNC is invoked using the new context and the parameters obtained from invocation of that function are stored in the cache to find in step 1920. The process continues in this manner until each of the nodes have been traversed and the needed caches have been defined in the appropriate context. This is generally illustrated by decision box 1928 and flow line 1962. If during the traversal of the scene graph, a USEFUNC node is encountered which has already been examined from the existing context, the existing context can be maintained and there is no need to create a new context for that USEFUNC node. This is illustrated by decision step 1916 and step 1932." (col. 28 line 55 – col 28 line 10.)

Applicant submits that Mayer in the above teaching indicates a method where executable opcodes in nodes are examined to determine of a new context is associated with that function. If a new context, such as a new color or shape or surface material is encountered, then a new context may be placed in cache. Once again, Applicant submits that

this does not teach the Claim 1 element of "requesting a search of the class path via the wrapper".

Applicant notes that Meyer does not teach or suggest requesting a search of a class using a wrapper because Meyer does not teach or suggest using any wrapper. Meyer uses cache to store information such as color, shape, surface material, texture, lighting, reflectivity, transformation and so on for the efficient execution of the opcodes. Applicant surmises that if a wrapper would be used, it would slow down execution and may render impotent the use of cache to speed up execution. Thus, Meyer fails to teach at least the Claim 1 element of "requesting a search of the class path via the wrapper" because the cache in Meyer is used for opcode argument data and Meyer fails to disclose a wrapper and thus fails to disclose requesting a search using a wrapper.

Applicant agrees with the Examiner on page 3 of the present Office Action that Meyer does not explicitly teach creating a wrapper for selected elements and a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements. However, Applicant respectfully disagrees that Yee comprehensively teaches the missing elements.

Yee teaches an extensible distributed enterprise application integration system (title). Yee states:

> "In general, in one aspect, the invention provides a modular application collaborator for providing inter-operability between applications including a plurality of connectors for communicating with a like plurality of applications and an interchange server." (Abstract)

Specifically concerning a wrapper, Yee teaches with respect to Figure 1(b):

> "For example, as shown in FIG. 1(b), the common scenario of enterprise resource planning (ERP) integration with custom legacy systems demands that the organization encapsulate complex processes properly within standard ERP implementations--not an easy thing to do. Many corporations choose to implement packaged applications for standard business processes such as inventory and order management. But

packaged applications are seldom used for vertical processes. For these purposes, the system 100 is ideal. It provides object interfaces for the ERP systems 22, 24, 26, 28, as well as wrapper-generation technology for linking to legacy systems 44, 48." Col. 14, lines 46-57).

Applicant notes that Yee teaches the existence of "wrapper-generation technology" for linking systems such as IMS (Information Management Systems) and CICS (Customer Information Control Systems) as shown in Figure 1B of Yee. Applicant respectfully submits Yee does not teach the Claim 1 element of "creating a wrapper for selected elements in the class path to provide a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements" because the wrapper of Yee is an integration of an IMS and CICS system with Enterprise Application Integration (EAI) System. The use of a wrapper in Yee is not equivalent to the use of the wrapper in Claim 1.

Concerning the citation to the teachings of col. 16 lines 54-67 and col 17, lines 1-22 on page 5 of the present Office Action, Applicant notes that Yee teaches:

> "As shown in FIG. 4(a), a typical intelligent agent-adapter 200 according to the present invention includes an agent component 210 and an adapter component 220. On one side of this architecture, the agent 210 conforms to a specified event and messaging model of the system 100. Adapter 220, on the other side of this agent-adapter architecture, uses a native application programming interface (API) 310 of a particular application resource 300, or other suitably published interface mechanism. Together, agent 210 and adapter 220 mediate differences in interface protocols and data structures to provide a uniform, normalized view of the business events that they publish and consume." (Col. 16 lines 54-65).

Here, Applicant notes that the intelligent adapter and the agent "mediate differences in interface protocols and data structures to provide a uniform, normalized view of the business events that they publish and consume". Applicant submits that this teaching is not related to the Claim 1 element of "creating a wrapper for selected elements in the class path

to provide a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements" because interface protocols, data structures, and normalized views of business events are not in the context of Claim 1.

> Yee does reference the use of APIs; as stated by Yee at col. 17 lines 66- col. 18 line 3: "Unlike past approaches to EAI, the foregoing agent-adapter architecture is extensible. It not only facilitates an ability to seamlessly accommodate changes to existing APIs, but it also continues to enable the use of those existing APIs with legacy systems." (col 17 line 66 – col. 18 line 3).

Applicant submits that these additional citations from Yee teach an API adapter that connects to legacy systems such as IMS an CICS. Yet, Yee still fails to teach anything in the context of the Claim 1 element of "creating a wrapper for selected elements in the class path to provide a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements." As such Yee fails to teach the elements missing from Meyer's teachings.

Accordingly, Applicant suggests that one of skill in the art would not use the teaching of Yee of a wrapper as a function to integrate a legacy system such as an IMS or a CICS with the Meyer teaching of a system to encode an animation into opcodes and transmit it to a users machine in separate loads for subsequent execution to arrive at the invention of Claim 1, where wrappers provide a level of indirection to APIs for different caches in conjunction with requesting and satisfying a search of a class path.

In brief review, there is no teaching or suggestion in Meyer that a wrapper be used at all. Consequently, there is not teaching or suggestion that an element such as "requesting a search of the class path via the wrapper" is disclosed in Meyer. Also, the "wrapper" of Yee is used to integrate legacy systems and is not clearly useable with the Meyer method of animation execution with a nanokernal after translating the animation to opcodes.

Applicant thus respectfully submits that the combination of Meyer and Yee does not teach or suggest the specific recitation in Claim 1 because all elements are not present in the two references. Specifically, the combination does not teach or suggest "creating a wrapper for selected elements in the class path to provide a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements and requesting a search of the class path via the wrapper" as recited in Claim 1. Thus, A prima facie case of obviousness under 35 U.S.C. §103(a) is not established per MPEP §2143.03 because the cited art does not teach all elements.

In addition to failure of the references to comprehensively teach all of the elements of Claim 1, Applicant can find no motivation to combine the teaching of an animation execution scheme using a nanokernal with opcode translation of the animation (as in Meyer) with a enterprise application integration system which can link legacy systems (as in Yee) to arrive at the invention of Claim 1. Without a cohesive motivation, found either in the references themselves or by one of skill in the art, a prima facie case of obviousness under 35 U.S.C. §103(a) is not established per MPEP §2142.

Since similar aspects of a wrapper providing a level of indirection are present in independent Claims 5, 10, 15, 17, 22, and 23, then Applicant respectfully traverses the current 35 U.S.C §103(a) rejection for the above stated reasons and submits that all pending independent Claims 1, 5, 10, 15, 17, 22 and 23 and their respective dependent patentably define over the cited art. Accordingly, Applicant respectfully requests reconsideration and withdraw of these claim rejections under 35 U.S.C. §103(a).

Claims 4, 7, 19-20, and 24 stand rejected pursuant to 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,243,856 to Meyer et al. in (Meyer) view of U.S. Patent No. 6,738,975 to Yee et al. (Yee) in further view of U.S. Patent No. 6,654,954 to Hicks. Applicant respectfully traverses the rejection based on the arguments above and the observation that Hicks also fails to teach at least the elements of "creating a wrapper for selected elements in the class path to provide a level of indirection from application

programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements and requesting a search of the class path via the wrapper". Therefore Claims 4, 7, 19-20, and 24 cannot be rendered obvious by the combination of Meyer, Yee, and Hicks. Accordingly, Applicant respectfully requests reconsideration and withdraw of the Claim 4, 7, 19-20, and 24 rejection under 35 U.S.C. §103(a) because these claims patentably define over the cited art.

Conclusion

Applicant respectfully requests reconsideration of the pending claims in light of the remarks presented above. An early Notice of Allowance for all pending claims is earnestly solicited because the un-amended pending claims patentably define over the cited art.

Respectfully Submitted,

Date: December 1, 2006

\Jerome G. Schaefer\

Jerome G. Schaefer
Registration No. 50,800

Woodcock Washburn LLP
One Liberty Place - 46th Floor
Philadelphia PA 19103
Telephone: (215) 568-3100
Facsimile: (215) 568-3439